JAVA 新闻APP 报告

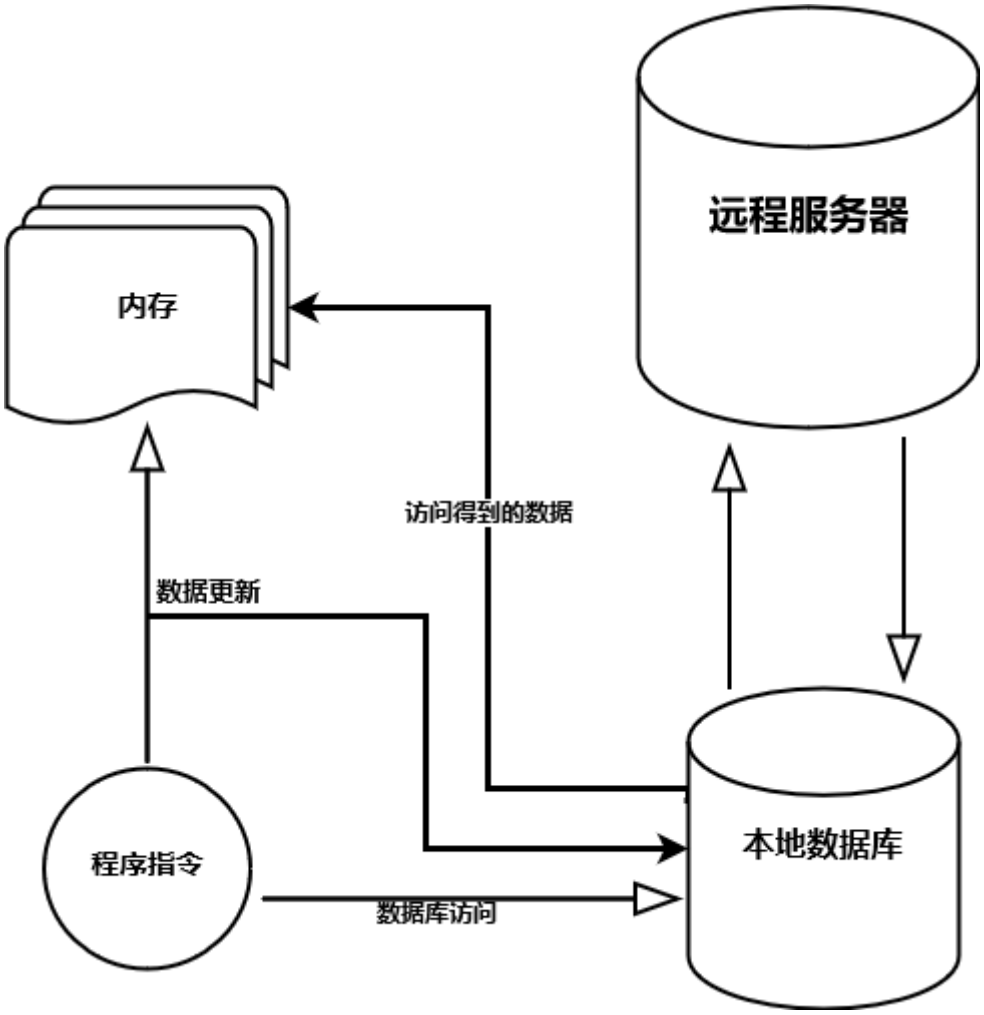
2017013588 计74 陈暐泽 2017013617 计76 彭皓

1 功能实现

功能	子功能	百分比（%）	是否实现
系统支持	要保证程序在安卓机上正常运行，测试过程中程序不崩溃	8	是
页面布局	布局合理，点击处理正确	8	是
分类列表	删除和添加操作	4	是
新闻列表	正确显示新闻列表的消息，布局和展示，点击进入新闻详情页面正确。	8	是
	实现新闻的本地存储，看过的新闻列表在离线的情况下也可以浏览。新闻是否看过的页面灰色标记。	8	是
	上拉获取更多新闻，下拉刷新最新新闻。	4	是
	显示新闻的来源和时间	4	是
	新闻关键词搜索，历史记录	4	是
分享收藏	使用微信、微博等SDK分享，新闻详情页面点击分享可以分享到常用的app，分享内容带有新闻摘要、URL和图片	4	是
	新闻详情页面点击收藏的添加和删除，实现收藏新闻的本地存储。收藏页的正确展示，点击可以进入新闻详情等	8	是
新闻推荐	根据用户看过的新闻推荐相关的新闻，参考今日头条等	8	是
	附加功能		
夜间模式	此功能主要为了减小应用在夜间对用户眼睛的光刺激，可以帮助用户在夜晚获得更好的用户体验	4	是
关键词屏蔽	此功能主要为了使得用户能够选择屏蔽自己不喜欢的话题（词语），获得更好地用户体验	4	是
登录功能	此功能主要为了让用户能够在云端储存自己的用户信息以及阅读兴趣。	8	是
评论功能	此功能为了让用户能够更方便地发表自己对新闻的看法，并看到其它用户的想法	8	是
发布功能	此功能为了让用户除了看新闻以外，还能自己发布信息、转发新闻，增强用户粘性	8	是
关注功能	此功能为了让用户还能够关注自己所感兴趣的用户，获得他们所发布和转发的新闻。	8	是

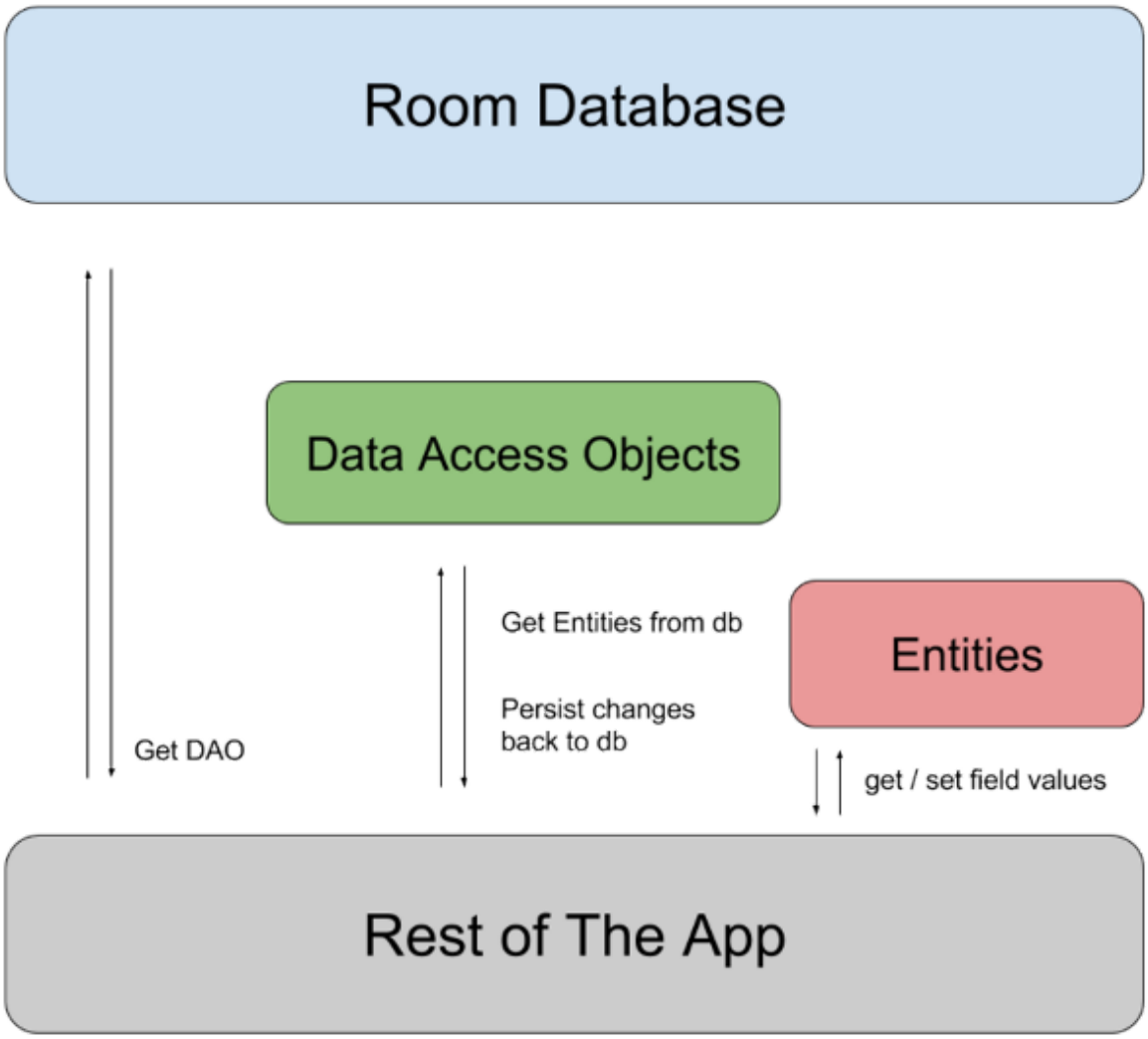
2 基础后端介绍

这部分主要介绍该app的后端基础——本地数据库和远程服务器。后端的框架是以这两个部分为基石建立起来的。 以下是整个后端的框架结构图。内存表示正在运行的Andriod程序在内存中维护的数据；数据库表示在本地外部存储器的数据，远程服务器表示在远程服务器的数据。



2.1 本地数据库

本地数据库使用Android自带的数据库——Room来实现。Room是google推出的用于做数据持久化保存的一个库。通过注释手段来实现一个抽象层，更数据库打交道，也是官方推荐的数据库。Room的基本框架如下图，关于Room的细节，在此不再进行冗余的介绍。



我在本地实现了两种数据库表单，分别存放两个不同类型的数据——新闻和用户。以下我会单独来介绍这两种数据库表单的结构以及主要的访问方法。

2.1.1 新闻表单

我为新闻类的数据构造了一种表单，在这个表单中的列包括newsID, keywords, image, title, content等新闻的各种必要信息。通过定义NewsDao接口实现程序和数据库的交互以及数据库的管理。在本次作业中，我为新闻类的数据创建了五张表单，用来分别存储访问过的新闻（离线状态也可以查看），用户访问的历史新闻，用户收藏的新闻，用户转发的新闻和用户自己发布的新闻。每个表单记录不同的新闻数据，方便管理和快速的访问操作。

2.1.2 用户表单

我为用户类的数据构造了一种表单，在这个表单中的列包括email, name, password, following, avatar等用户的各种必要信息。通过定义userDao接口实现程序和数据库的交互以及数据库的管理。在本次作业中，我为用户类的数据创建了一张表单，用来存储注册过的用户。这个表单存在的意义是可以在本地方便的得到用户的信息，从而方便的实现诸如新闻评论，关注等之类的操作。

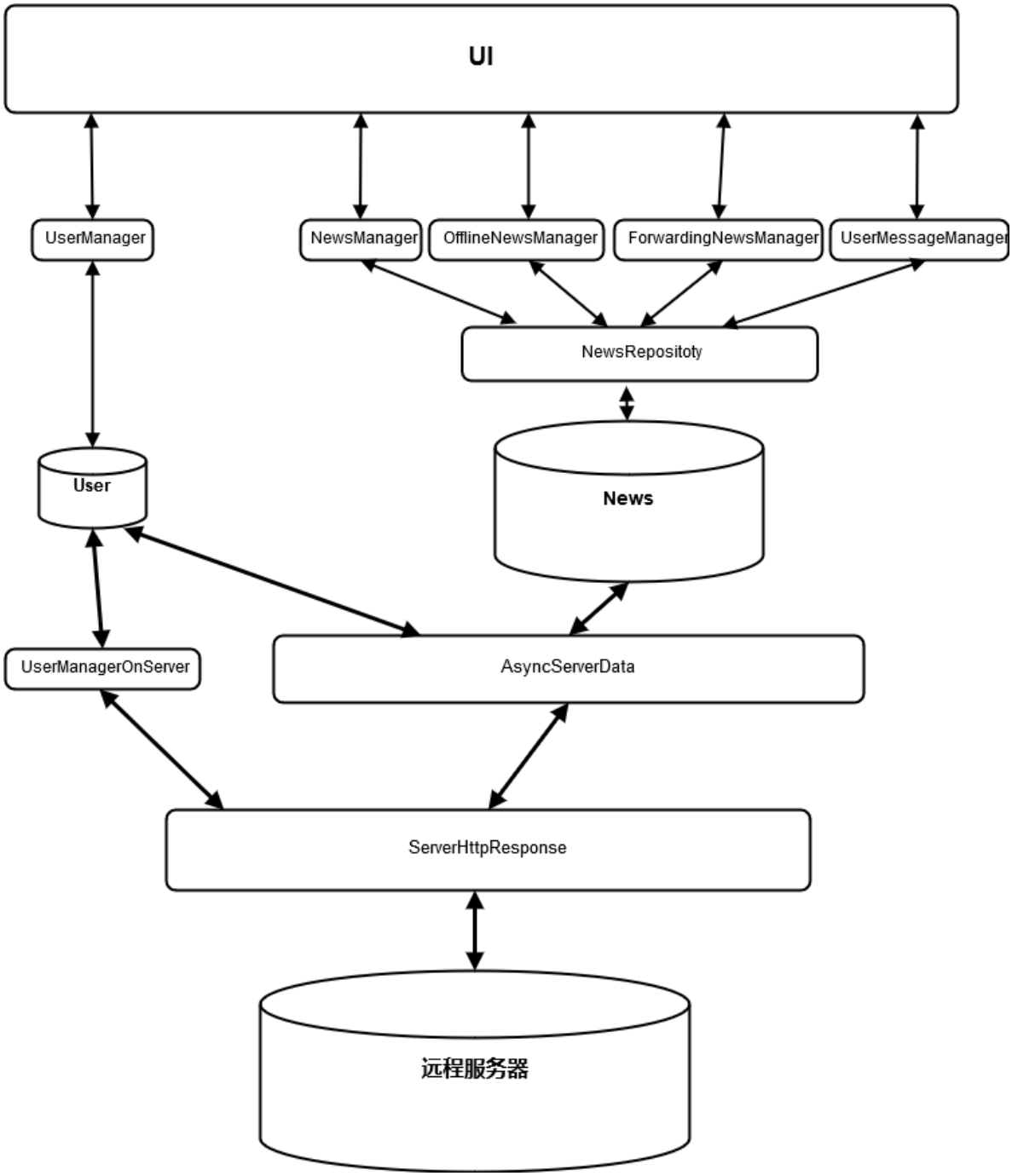
2.2 远程服务器

远程服务器使用Django进行搭建。Django是python的一个web服务器框架，方便搭建，方便管理。但是原生Django的并发处理能力比较差，对于多条请求，会出现阻塞的情况，但是对于本次作业而言，这个远程服务器已经足够。 在服务器端和本地数据库一样，我为新闻类的数据创建了四张表单（没有为离线新闻创建），创建了一个用户类表单，同时还创建了专门用来存储图片的表单。当来到请求之后，

django会执行相应的函数，进行服务器和数据库的交互。从上图可以看出来，为了方便同步以及清晰的数据管理，我只允许数据库和远程服务器之间的交互，内存是不能直接和远程服务器进行交互的。

2.3 后端的Java类框架

下图是整个后端类的总览。



以下列表是后端主要功能类的介绍：

- News类：此类的功能是记录新闻数据的各种必要信息。
- NewsRepository类：此类的功能是与News数据库进行交互，对数据库操作提供了异步执行的封装。在执行数据库操作之前，此类的每个方法都会构造一个AsyncTask类的子类实现任务的异步执行。
- NewsManager类：此类的功能是管理historyNews和collectionNews数据库表单，也即管理用户的浏览历史和收藏。此类是单例模式。
- OfflineNewsManager类：此类的功能是管理离线新闻，也即看过的新闻，保证离线状态也可以查看。同时提供了图片缓存的方法，可以将新闻的图片缓存到本地。此类是单例模式。
- ForwardingNewsManager类：此类的功能是管理用户转发的新闻。此类是单例模式。
- UserMessageNewsManager类：此类的功能是管理用户自己发布的新闻。此类是单例模式。
- User类：此类的功能是记录用户自身的各种必要信息。比如关注者，头像等信息。

- UserManager类：此类的功能是管理本地数据库中的用户。构造AsyncTask的子类实现任务的异步执行。此类是单例模式。
- ServerHttpResponse类：此类的功能是通过http协议实现与远程服务器的通信，此类通过构造AsyncTask类的子类实现任务的异步执行。
- AsyncServerData类：此类的功能是上传数据和下载数，里面有四个方法：ayncNewsToServer, ayncNewsFromServer, asyncUserToServer, asyncUserFromServer 作用分别是上传用户的新闻数据，下载用户的新闻数据，上传用户的信息，下载用户的信息。此类是单例模式。
- UserManagerOnServer类：此类的功能是实现用户在远程服务器的注册，登陆和退出操作，同时执行必要的同步操作。此类是单例模式。

3 具体实现

3.1 应用初始界面

打开APP后进入MainActivity中，在本应用的设计中，这只作为一个跳转页，展示应用的封面，如上图。设置一个三秒的计时器，到时后有两种可能性：

1. 手机第一次打开本应用，进入动画界面。
2. 非第一次打开，直接进入新闻列表。

3.2 开篇动画

使用`TextSurface`包进行绘制。动画结束后，自动渐变进入新闻列表。但由于这个包5年前就停止维护了，导致某些效果无法使用，调试了很久。

3.3 使用Fragment

由于新闻分类显示、历史新闻、收藏新闻、搜索新闻都是类似的，都是对新闻进行逐条显示，所以考虑使用`Fragment`在这几个功能之间进行切换。所以这些界面都设置为了Fragment。切换时只要`commit`上一个新的`Fragment`就可以了，不用反复打开关闭`Activity`。

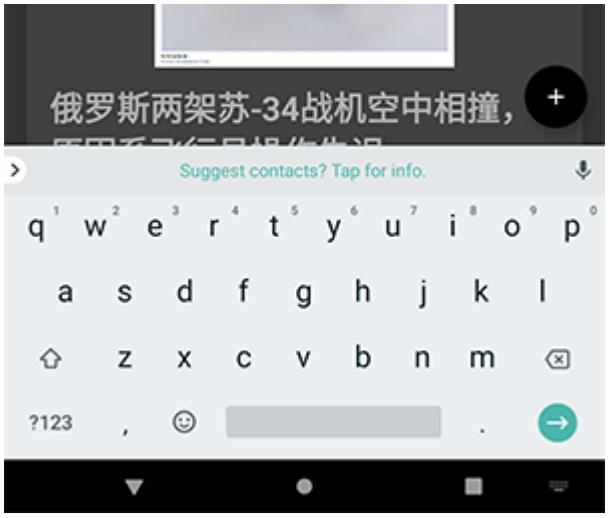
3.3.1 新闻分类显示的Fragment

新闻列表的主体由三部分组成：

1. 一个可以随着下滑而隐藏的标题栏
2. 一个可以左右滑动的类别列表
3. 一个可以上下滑动的新闻列表


11:31
LTE
News TODAY
推荐 关注 娱乐 军事 教育 文化 健康 财
海南省委常委、海口市委书记张琦接受调查
中新网9月6日电 据中央纪委国家监委网站消息，海南省委常委、海口市委书记张琦涉嫌严重违纪违法，目前正接受中央纪委国家监委纪律审查和监察调查
市委书记
Two Su-34 jets collide in Russia due to pilots' error — source
俄罗斯两架苏-34战机空中相撞，原因系飞行员操作失误
原标题：俄罗斯两架苏-34战机空中相撞，原因系飞行员操作失误【环球网报道 记者 温家越】当地时间9月6日早些时候，俄罗斯两架苏-34战机在利佩茨克地区训练时发生空中相撞事故
飞行员操作


11:33
LTE
Type key words you want to query
推荐 关注 娱乐 军事 教育 文化 健康 财
海南省委常委、海口市委书记张琦接受调查
中新网9月6日电 据中央纪委国家监委网站消息，海南省委常委、海口市委书记张琦涉嫌严重违纪违法，目前正接受中央纪委国家监委纪律审查和监察调查
市委书记
Two Su-34 jets collide in Russia due to pilots' error — source



具体如上左图所示。具体实现为

标题栏

随着用户上滑，标题栏将逐渐向上隐藏，一旦下滑，标题栏将再次出现。这一实现依靠的是Google官方提供的`CollapsingToolbarLayout`。在这一layout中放入一个`ImageView`用来显示标题栏图片，然后在右侧放置一个`SearchView`。这一`SearchView`可以被点击，点击后整个标题栏都将称为输入搜索关键词的输入框，如上右图所示(此图是在夜间模式下的截图)

分类列表

分类列表使用一个`RecyclerView`来实现，以支持左右滑动。在对应的`Adapter`中维护一个`lastClicked`变量，记录上次点击了哪个类（初始为null），然后为每一个类别绑定一个`onClickListener`，只要被点击了，就将`lastClicked`对应的类别背景变黑，高亮当前类别背景，并更新`lastClicked`变量，最后按照类别请求。

右侧有一个按钮，按下之后可以进入到对类别进行增加/删除以及排序操作的界面，如下左图所示


11:33
我的频道
点击删除频道，长按拖动调整顺序
推荐
关注
娱乐
军事
教育
体育
汽车
科技
社会
隐藏的频道
点击添加频道
+健康
+文化
+财经


9:03
News TODAY
推荐 关注 娱乐 军事 教育 文化 健康 财
原创 20支导弹部队随时恭候 俄军总长发出最强硬宣言：又来3架算个啥?
原标题：20支导弹部队随时恭候 俄军总长发出最强硬宣言：又来3架算个啥
战略轰炸机

上下各有一个自定义的`GridLayout`，实现了拖动时半透明的功能，以及实时插入所在位置的效果。在`我的频道`中点击对应的频道，可以删除，在`隐藏的频道`中点击被删除的分类，可以恢复，长按可以进行分类的拖动排序。

后期感到使用`RecyclerView`来实现分类列表实在不算是一个好的选择。似乎可以使用和Tab相关的一些Layout来进行实现，似乎会方便很多，不用自己进行选择高亮等操作。

新闻列表

使用的是`RecyclerView`。其中每一个`View`使用的是在`CardView`。在每个`CardView`中有一个位于顶部的`ImageView`，用于显示新闻的第一张图片（如果有的话），如果新闻没有图片，则将`ImageView`的`visibility`设置为`GONE`。下方是加粗的标题文字以及新闻中的第一句话（如果第一句话长度长于100，则截断加省略号）。左下方用一个方框将关键词装入，右下方包含有一个发布者头像（只在用户转发的新闻或是用户自己发布的消息中显示）、转发按钮、一个收藏按钮（收藏后被点亮）。

整个`View`被设置了一个`onClickListener`，一旦被点击，则可以进入对应新闻的详情页。

下拉刷新以及上拉加载的实现基于`SmartRefreshLayout`，这是一个第三方包，可以方便地实现下拉刷新和上拉加载的逻辑。

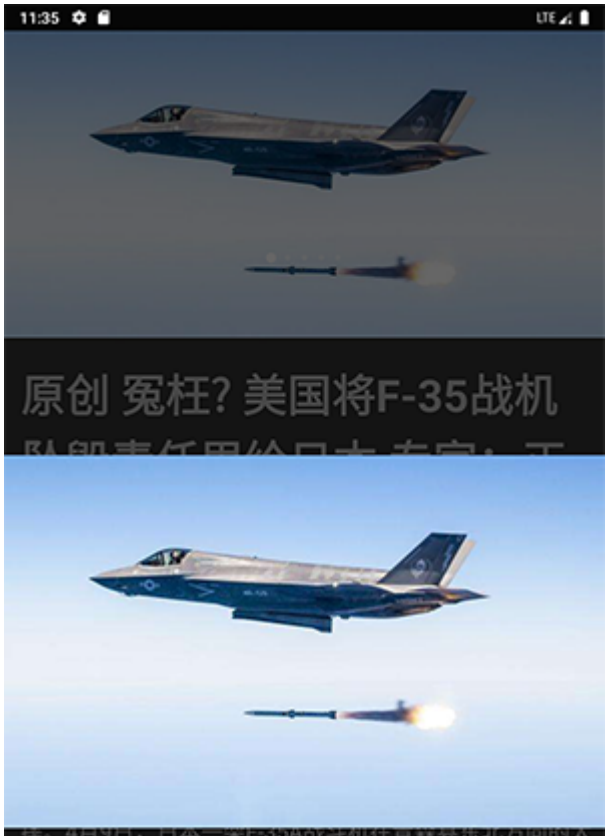
每一个新闻列表中的项可见上右图

发布按钮

可以注意到在页面右下角存在一个圆形浮动按钮，基于Google官方提供的`FloatingActionButton`实现，后期会进行介绍。

3.4 新闻详情页

由于不知道新闻中到底会有几张图片，所以很难动态地在新闻文字当中插入图片。因此我们的想法就是在页顶设置一个`ViewPager`，可以左右滑动查看新闻包含的所有图片，由于要让整个布局随着图片的大小调整高度过于耗费资源，而且容易出现bug，所以采用固定高度的策略。这样会造成某些图片无法显示完全，因此在每一张图片上设置了一个`onClickListener`来监听点击事件，一旦被点击，则通过一个没有选项的对话框展示图片，如下图所示。

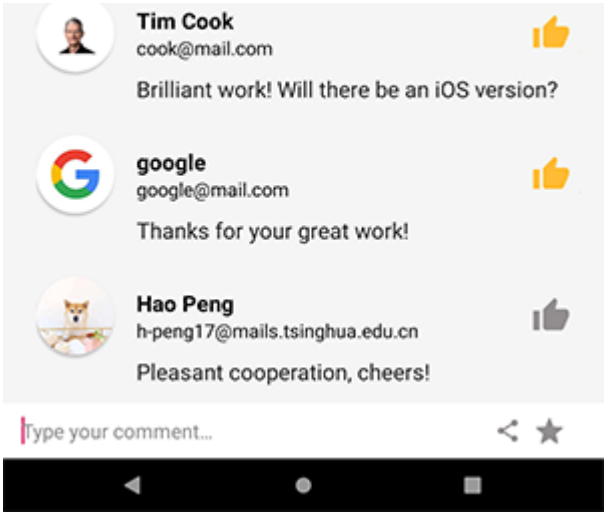


We are now pleased to announce the official relase of our News Today APP!

google@mail.com 2019-09-07 08:53:44

Bill Gates
billGates@mail.com

Excellent!



底部存在一个可随着滑动隐藏和出现的底栏，其中可以输入评论，也可以在其中进行分享和收藏操作。右上图中我们给出了一个展示评论的例子（假的新闻）。

3.5 历史新闻

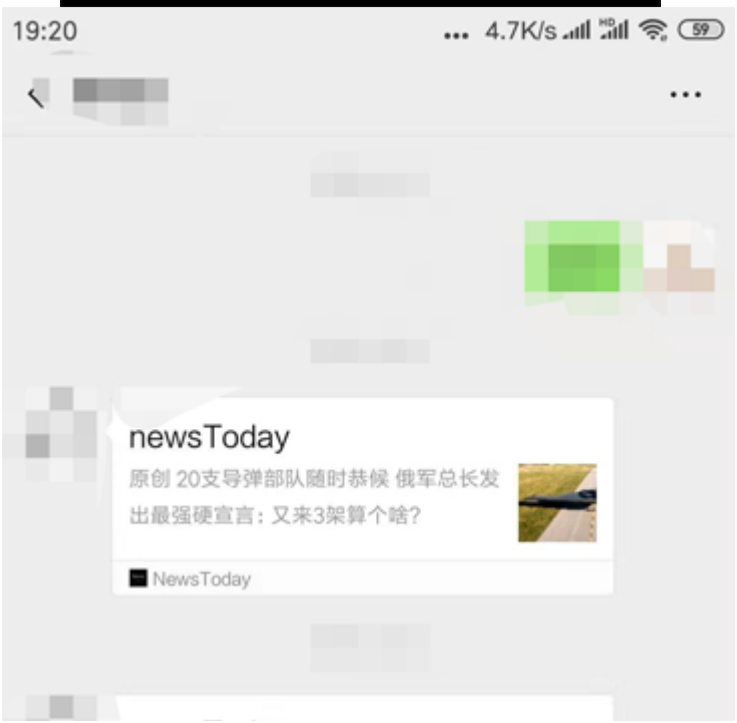
在新闻列表中监听点击事件，点击新闻的同时，除了打开新闻详情页以外，还在后端将该新闻加入到数据库中，同时将会把新闻标题置为灰色。 后端会通过NewsManager类对历史新闻进行管理，可以将历史新闻加入数据库。同时为了前端判断一个新闻是否在是历史新闻中，后端专门在内存里面维护了一个ArraySet<String>用来记录在history中的新闻newsID, 同时将该ArraySet和数据库实时同步，确保逻辑的正确性。 对于历史新闻，清除全部历史的选项，当选择清楚全部历史时，后端会清空历史新闻的表单，并实时同步在内存中。

3.6 收藏新闻

在新闻列表中对收藏按钮监听点击事件。应用中的收藏按钮都基于ShineButton实现，它用简洁的方式提供了优秀的交互效果。用户对按钮进行点击后，将有动画反馈。与此同时，检查点击后的状态来判断是删除收藏还是添加收藏，并更新到后端数据库中。后端同样通过NewsManager类管理收藏数据表单，根据前端的动作实现收藏新闻表单的添加或者删除。同时，在内存中维护一个ArraySet<String>来记录收藏新闻的newsID，并且和数据库实时同步。

3.7 分享新闻

在新闻列表中对分享按钮监听点击事件。在点击分享按钮之后，底部会出现菜单，这个bottom dialog是基于BottomSheet实现。他提供了简洁的画面，并且可以很方便的设置点击监听事件。具体效果可见下图



由于对微博的app申请没有获得批准，所以只实现了分享到微信的功能。微信分享功能靠WechaShareManager类来实现，可以实现分享到会话或者分享到朋友圈。可以分享问题，图片和URL，具体实现效果如上图。 微博申请被驳回

3.8 搜索新闻

通过`Android`官方提供的`SearchView`来完成搜索功能的配置。具体步骤与官网上的教程一致。只不过在`Search Activity`当中执行的是将`query`作为查询关键词，输入到`api`中，获得返回的新闻。通过继承SearchRecentSuggestionsProvider 类构造一个记录搜索历史的类，并且通过这个provider自动为SearchView提供内容，这个父类也是`Android`官方提供的。

3.9 关键词屏蔽

用户输入的关键词会首先存放到内存当中，并存放到一个文件当中。 在用户修改关键词的时候，会将关键词更新到对应的文件当中。在上传用户数据的时候存储屏蔽词的文件会同步到服务器上。

3.9.1 设计思路

「屏蔽」所引起的联想似乎都不太美好：于应用管理者而言，屏蔽虽然在一定程度上提高了内容质量的下限，但可能同时也意味着对某些潜在价值观的抵制；于使用者而言，屏蔽是最简便的“眼不见为净”的践行方式，但可能同时也意味着信息茧房的不断收束。

所以前端在这一功能的UI设计上，将一句问句与所要填写的屏蔽词结合，希望能够引起用户的思考"屏蔽这个词，不再接受这个词的任何新闻，真的好吗？"。以噪声的波形图作为背景，辅以故障风的文字，试图营造出一个并不舒适的环境，并隐式地传递给用户设计者的立场。

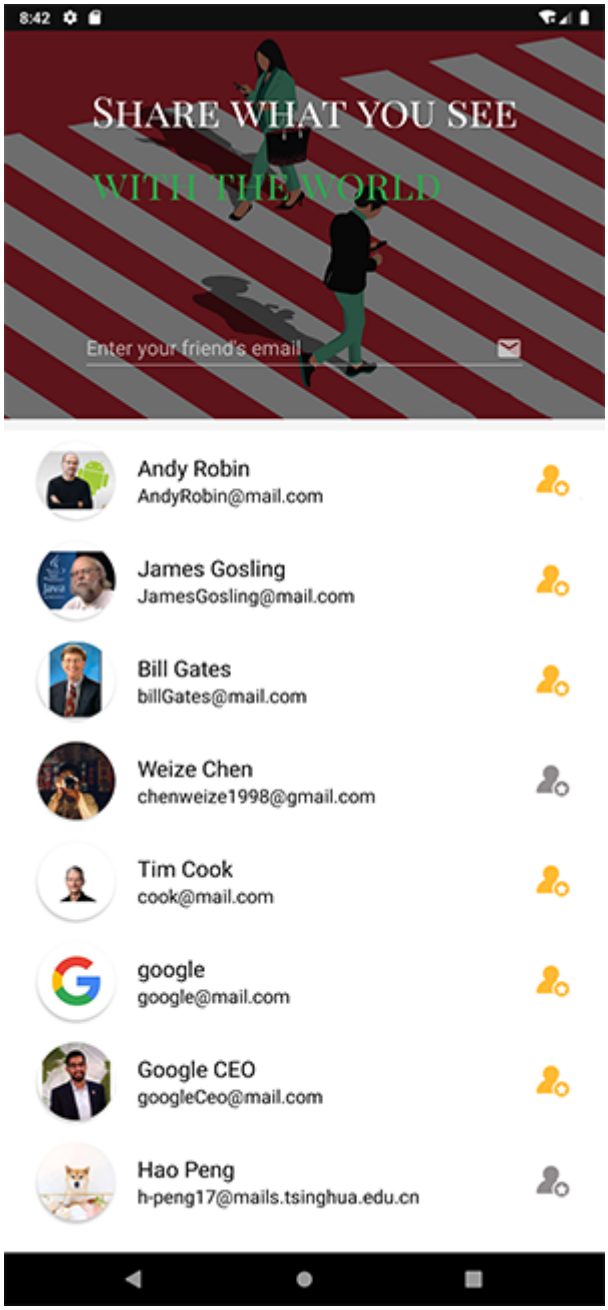
3.9.2 实现

背景只是贴图，耳朵也是一个`ImageView`。运用`FrameLayout`可以将三个`TextView`进行叠加，分别设置不同的垂直方向偏移量以及不同的颜色，即可以最简单的方式形成故障风的文字。

Filter Words的显示复用了之前在分类排序中的`GridLayout`。

3.10 查找好友

查找好友功能的实现，主要是通过UserManager类。用户输入被查找人的email, 后端会调用UserManager中的getUserByEmail方法从数据库中检索出这个用户的全部信息（如头像，昵称等），然后返回一个User对象，前端需要展示这个user的很多信息，所以直接返回一个User对象是方便的，合理的。得到的用户信息会被展示到空白位置。具体效果如下图



3.10.1 设计思路

既然要做社交，那么就应该鼓励用户使用社交功能多与其它用户进行互动交流分享。所以选图时的一种思路是找到很多人在一起social的场景；一种思路是突出独身与社交的冲突；还有一种思路是反应当下人们过分沉浸于自我的世界中而疏于与身边人交互的状态。色彩上，应用黑白灰的主色调并不太适合社交的活跃性，因此希望是鲜明或存在明显对比冲突的。最后选择了最后一种思路，挑了这样一张图，两个人擦肩而过却都低头看着手机，同时还存在明显的色彩冲突。蒙上一层黑色的半透明层就可以作为底图了。

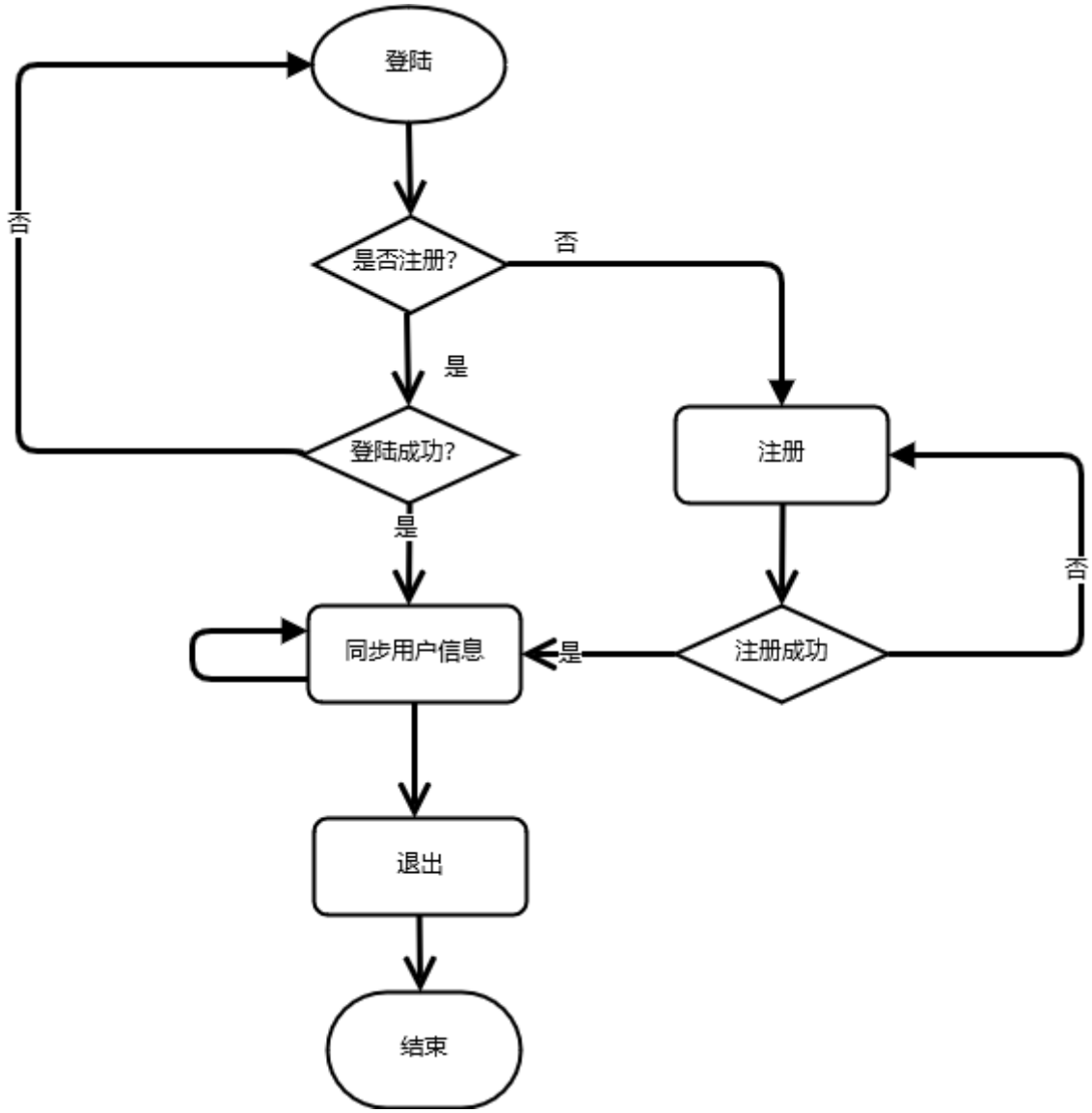
在Android Studio中可以选择Google Font上的字体，原字体过于沉闷，所以选择了更具有灵性的一个字体作为标题字体。

3.10.2 实现

主体是下方的一个`RecyclerView`。由于Android没有自带圆形的`ImageView`，因此其实使用了一个`CardView`，设置了圆角，并令其半径就是`CardView`宽度的一半，再在里面放上`ImageView`就成了一个圆形头像。为每个人的头像设置了点击事件的监听，点击后可以进入此人的个人主页（展示此人转发的新闻和发布的消息）（没有截图）。

3.11 登陆

我们为实现了用户的注册，登陆和退出功能。当用户输入消息后，点击注册，后端会调用UserManagerOnServer类中的signUp方法，与远程服务器进行通信，远程服务器判断这个注册是否有效，如果有效，则修改服务器的状态，并且返回成功的信息；如果无效，则直接返回失败的信息。在调用signUp函数的同时，如果返回成功，则会在本地创建一个User类的对象，并且放到user的数据库表单之中。当用户输入消息后，点击登陆，后端会调用UserManagerOnServer类中的signIn方法，与远程服务器进行通信，远程服务器会判断这个登陆是不是有效，如果有效，则修改服务器的状态，并返回成功的信息；如果无效，则直接返回失败的信息。在调用signIn函数的同时，如果返回成功，则创建一个User类的对象，并从服务器将关于这个user的信息全部下载下来，并且存储到本地数据库之中。在用户登陆或者注册成功之后，如果想退出，则直接点击log out, 后端会首先将这个用户的全部数据自动上传到服务器，并且更改服务器的状态，然后将本地的关于这个用户的数据删除，节省空间。 为了使叙述更加清楚，下面流程图展示了一个user的生命周期：



上图是登陆界面，登陆界面挺简单的，两句Slogan加上传统的登陆信息就构成了这个界面。`Get to Know the World`表明了我们希望我们的应用能够帮助用户了解到世界上发生的各种各样的事件，而`News are temporary, pride is forever`中的`pride`更多的想指我们所做出这个APP所产生的成就感。

Slogan下的一条红横线其实是一个高度为1dp的View，不是实际的线。三个输入框是官方提供的`TextInputEditText`，下面两个按钮就是普通的`Button`

3.12 发布消息

我们实现了用户发布消息的功能。用户发布的消息同样用News类来表示，不同的是里面的publisher属性改成用户自己的邮箱。点击发布按钮，可以进入编辑发布消息的界面。输入title, 和 content, 并且可以选择图片，图片最多选择9张。为了和原本的News统一，也为了存取方便，发布的图片会首先上传到远程服务器，然后得图片的url, 存储在News类的image域中。对于发布的消息，通过

UserMessageManager类进行管理和同步。

3.13.1 设计思路

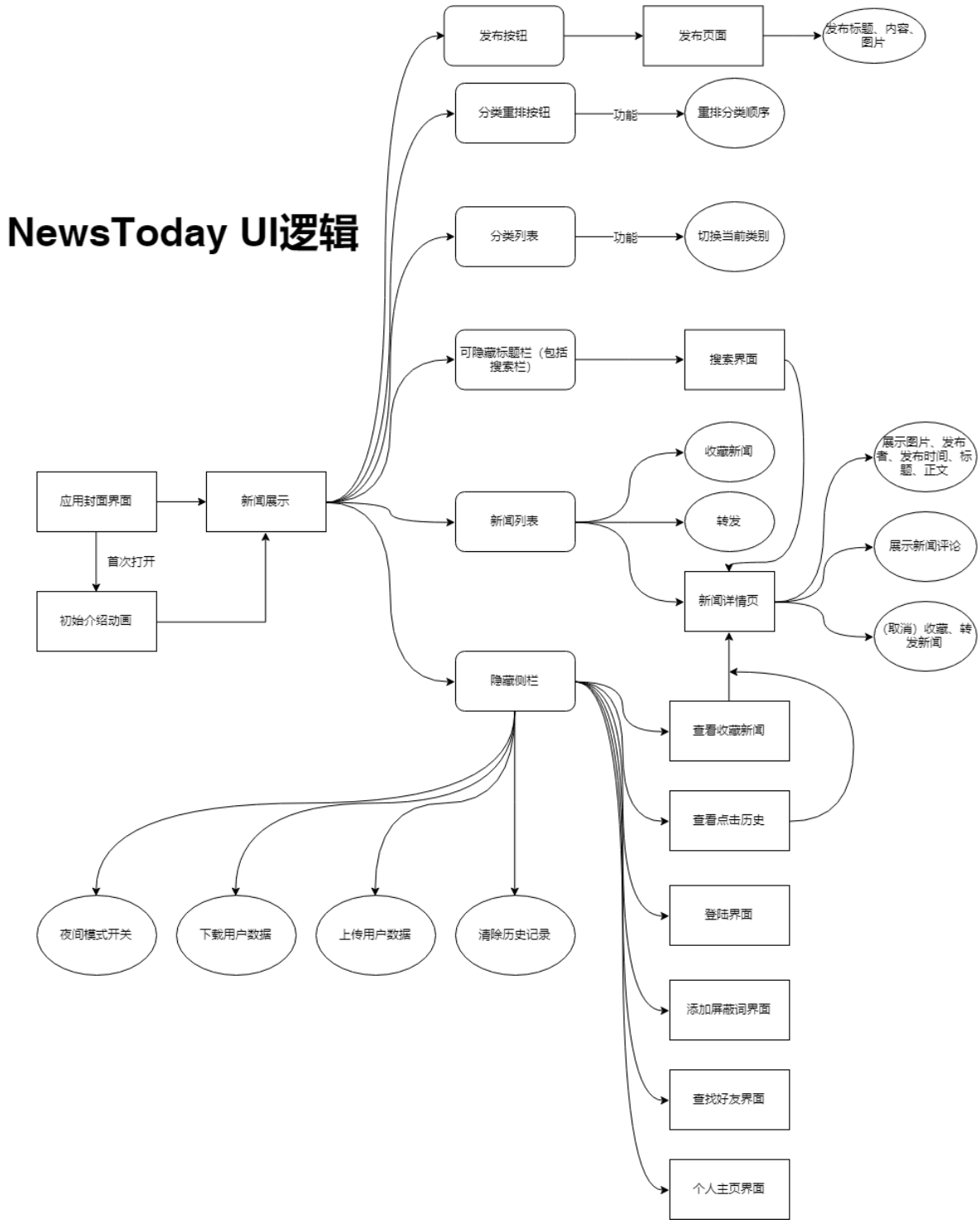
对于用户自己发布信息我们是持鼓励态度的，毕竟社交功能是我们这个APP一个很重要的功能。在完成这一页面的设计时，一个模板当然就是每天都在使用的微信（添加图片的+图标就是仿的微信）。但由于我们是一个新闻APP，发布的消息希望能够有标题和正文，所以最后分了两栏分别输入。考虑到界面的单调性，决定在上方加一个能够自动expand、自动hide的标题栏。选图的标准是希望能够反应我们对于用户发表动态的鼓励，并且在风格色调上与之前布局相似的查找好友相仿。最终选择了Andy Worhal以及他的那现在被自媒体用烂了的观点

> In the future, everybody will be world famous for fifteen minutes. Andy Warhol (1928-1987)

3.13.2 实现

自动伸缩的标题栏采用的还是之前使用过的`CollapsingToolbarLayout`。在右上角单独加了一个`Button`用于监听点击的发布事件。下方两个`EditText`，一个由`GridLayoutManager`作为LayoutManager的`RecyclerView`，用于显示图片的网格。选择图片时使用了Matisse包，选择后，使用Luban进行图片压缩。

3.14 UI逻辑



4. 总结与心得

前端：陈暐泽

本次大作业我主要负责完成前端工作。前端的工作相比起后端对数据繁杂的处理而言，是相对清晰而简单的，出现的bug也不如后端的bug那样难调，但写前端仍然不是一件容易的事。

前端面临的**第一个考验是各种Layout的特性**刚上手的时候几乎是崩溃的，尤其是安卓官方现在十分推荐使用的`ConstraintLayout`，它的某些XML特性并不会按照该特性的名字所暗示的那样起作用。例如`layout_constraintTop_toTopOf`和`layout_constraintBottom_toBottomOf`，如果都指定为`parent`，那么其实它可能会垂直居中而非拉伸扩展。又例如处在`ConstraintLayout`中的`RecyclerView`，将宽指定为`match_parent`而长度指定为`match_parent`，则`RecyclerView`很可能将覆盖整个屏幕，而长度指定为`wrap_content`，则很可能小时不见；所以令人惊讶的是在`ConstraintLayout`中的`RecyclerView`应该指定为`0dp`，让它自己尽可能在可用的空间中决定自己的

长度。总之调各个界面的Layout的时候，会面临各种玄学bug，以及各种自己难以实现的想法，在Layout这一层面上就遇到了不少问题。

第二个考验是界面的审美设计。要一个程序员既能编得一手好程序，又能有尚可的审美和一脑子设计想法，这个要求实在有点高。但原生自带的界面实在丑陋原始得令人感觉自己仿佛回到了十余年前那个诺基亚还是老大哥、3G还是个传说、北京申奥刚刚成功的信息远古时代。虽然这可能能够成功唤醒用户的怀旧思念之情，但这毕竟不是我们想要的应用界面。所以一个巨大的问题就是如何才能将界面设计得不落俗套、简洁美观，如果可以的话，再偷偷夹带点私货，藏点自己的想法和个人风格。

国内新闻APP像今日头条等，大多采用浅色为主色调。但我们在配色上并没有进行仿制，而是反其道而行之，选择了黑白灰这样的朴素的颜色为主色调。我们所认为的好新闻，应该是客观理性的、经过沉淀的，应该是不张扬不浮躁，文字素朴却有力的。这样看来，黑色的沉稳和白色的力度，最能体现我们这种想法。（好吧其实黑白灰的配色还有一个好处就是转换成夜间模式的时候要改的颜色也少一点）（好吧但是而且我们也就是调助教提供的API，展示的新闻大杂烩里哪里能够有什么我们的理念呢 :)。我们在很多地方都展现了自己对新闻的想法。比如应用启动时的封面页，底端文字“Informative · Inspiring”，以及我们在登录页面上的Slogan "News are temporary, pride is forever."，以及屏蔽词添加的界面的设计。

设计过程中难免需要参考其它的APP。新闻列表界面的设计上参考了`轻芒杂志`，采用了卡片的形式进行新闻简略的展示；夜间模式的配色参考了`飞地`；发布界面部分参考了`微信`；分类界面参考了`今日头条`。另外一些界面的设计中，最后也确实夹带了一些自己的私货，但由于这不是一个真正发布的产品，还是可以任由自己任性一把，简单粗糙地过一下设计的瘾。

但前端也不只是对界面的设计工作：页面之间的转移逻辑，信息保留传递，不同组件之间的交流互动，都是需要代码来保证实现的。而在这过程中的bug也没少出。甚至由于一开始上手时没有经验，没有采用fragment，而是采用了一堆的Activity，最终不得已进行了一次大型的重构这样极大程度考验重构和debug能力，同时耗费精力的工作。在使用GitHub上各种不同的库时，还要时常由于库中代码四五年未维护而无法与当前框架适配而自己对他们的代码进行重写。工作量可以说并不太小。

后端提供接口之完备是我能够安心完成自己前端工作的基石。彭皓自己完成了所有数据库以及服务器的配置工作，在我提出接口要求之后也能高效的完成相关代码的编写，极大提高了我们APP的开发效率，可以说没有他，我们的APP就只是能是一个空有其表、华(?)而不实的应用。

后端：彭皓

这次大作业遇到的Bug很多，付出的时间很多，最终的收获很多。这次大我主要实现后端部分。对于一个app, 后端的鲁棒性和健壮性是及其重要的，很多时候程序的崩溃问题都出现在后端。所以一开始我就先想好架构，争取用一个好的架构，来提高程序的鲁棒性。在刚开始写的时候后端的问题不是很多，无非就是接受新闻服务器的数据，提供给前端，并进行相关的数据库等操作。后端真正的难点和bug出现在了加了服务器之后。我们想实现服务器，通过服务器来进行用户数据的远程存储，可以使用户在不同的设备上依然可以获得自己的数据，这也是现在很多app的必需。搭建服务器是比较简单的，比较顺利，通过各种I/O编程，比较顺利的实现了本地数据和远程服务器的通信。经过测试，效果也是ok的。

但正当我们开心的时候，却出现了一个非常奇怪的bug。`protocolException: unexcepted end of stream`，起初以为这个是网络的问题，后来在仔细检查之后发现并不是网络的问题。然后开始到网站上搜索有没有遇到一样的情况，结果大家的解决办法都没有成功，后来我就怀疑是多线程的问题，经过翻来覆去的修改，发现也不是多线程的问题，就这样debug了一天一夜，还是没能找出这个bug，当时的内心是很绝望的，虽然离ddl还有10天，但是这个bug着实带来了很多困扰，在最后，我把自己写的服务器放到了一个专用的linux服务器上，这个bug就消失了，再也没有出现过，后来我们猜测这可能和我自己电脑有关，connection需要时不时的reset, 导致了这个迷惑的Bug。遇到的第二个迷惑的大bug就是在测试微信分享的时候，无论如何都无法拉起微信，并且提示签名不对，也是找了好久，也没找到这个Bug由什么引发的，后来换了一台手机测试，就没事了。其实这个Bug主要是因为微信缓存了之前的app的签名，所以你在设置新签名之后，微信还是用的之前的旧签名，这就造成了签名的不统一。以上是我遇到的两个超级玄学的bug，令我印象深刻。

除了以上两个Bug，其余的主要Bug出现在社交功能。我们在实现社交功能的时候，还要有新闻的评论功能，好友关注功能等等，这些逻辑的实现，是需要数据的同步的。比如一个用户评论了一个新闻，那个这个新闻在展示的时候就需要展示被评论的新闻，所以这个新闻如果在数据库，那么他就要被更新，如果这个新闻还在远程的服务器上，那么这个新闻还要在远程服务器上更新。而这个社交功能，难点就在于更新同步，如何同步数据，何时同步数据都是一个需要谨慎思考和解决的问题。后来我想到的办法就是每次更新完内存中的内容之后，都去数据库更新数据，并且更新到远程服务器；每次需要获取数据的时候，先从本地的数据库进行查询获取，而不是直接从新闻服务器拉取数据；在更新数据库的时候，首先要查看数据是否有可能被覆盖等等。总之，实现社交功能比实现新闻功能要难很多，但是梳理好逻辑之后，实现起来也不是很复杂。

这个大作业是进清华以来写的最认真的一个大作业，构架了一个良好的框架，写起来也比较得心应手，所以框架很重要，是后端的基石。从写这个大作业之后，几乎每天都会写到很晚，尤其是最后一周。我和陈暐泽一起开发了三周，330次commits, 上万行代码，最终能够写出来一个功能比较健全的app，很开心。最后，感谢合作者陈暐泽的carry, 陈暐泽为这个app做出了最重要的贡献，使我们的app具有优美的前端，很好的表现力。